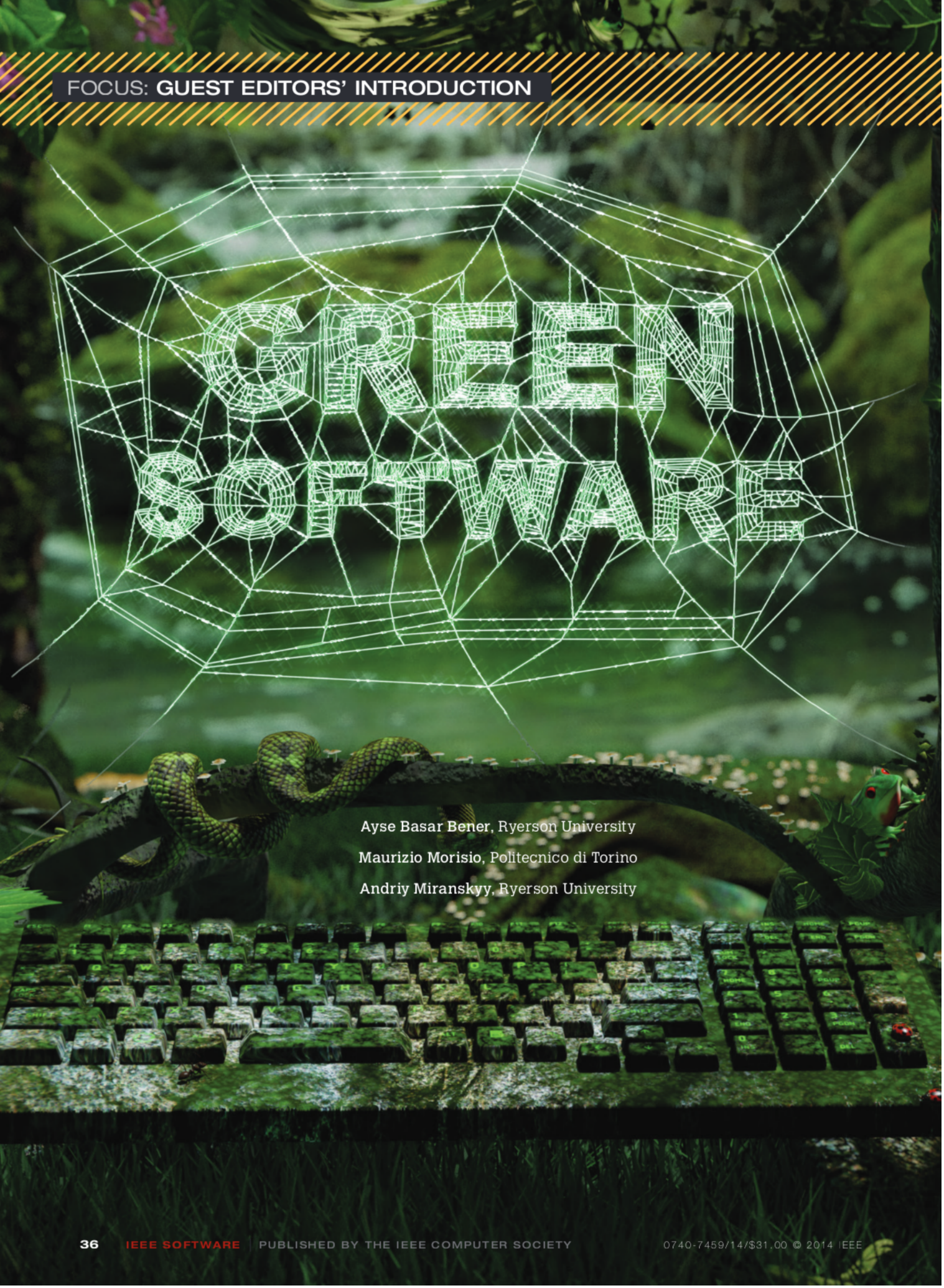


**Ayse Basar Bener**, Ryerson University
**Maurizio Morisio**, Politecnico di Torino
**Andriy Miranskyy**, Ryerson University


0740-7459/14/$31.00 © 2014 IEEE

**GREEN IT** is the study and practice of using computing resources efficiently to reduce negative impacts on the environment. Green IT is applicable to various high-tech domains, such as datacenters, mobile computing, and embedded systems. Recently, global carbon dioxide emissions reached 9.1 billion tons, the highest level in human history—49 percent higher than in 1990 (the Kyoto reference year).[1] At least 2 percent of global carbon dioxide emissions can be attributed to IT systems, and further increases are expected as new IT systems are deployed daily. Therefore, reducing the energy consumption and related carbon dioxide emission of IT systems is very important.

## Stepping Up

Most studies and regulatory controls focus on hardware-related measurement, analysis, and control for energy consumption. However, all forms of hardware include significant software components. Although software systems don't consume energy directly, they affect hardware utilization, leading to indirect energy consumption. Therefore, it's important to engineer software to optimize its energy consumption. The software engineering research domain has recently been paying attention to sustainability, as the increased number of publications, empirical studies, and conferences on the topic demonstrate.

Greening in software aims to reduce the environmental impact caused by the software itself. Therefore, greenness in software is an emerging quality attribute that must be taken into account. Software companies need to deal with the conflict between customer pressure (new functional requirements and a high level of quality) and being as environmentally friendly as possible.

As software continues to affect all aspects of our lives in ever-changing forms, leveraging existing systems is a challenging task for many companies. Keeping software available on demand with a high quality of service (with respect to user requirements) creates a conflict in terms of software energy consumption. As the new features are implemented, they may increase the level of energy consumption, making it difficult to maintain environmentally friendly software.

On the other hand, software systems can play a proactive role in saving energy by providing feedback about the way they consume resources. This could lead to changes in people's behavior to make greener processes. Building green software systems has implications for environmental awareness and behavioral changes that might contribute to the building of smart communities and cities.

## In This Issue

For this special issue of *IEEE Software*, we sought articles that cover various aspects of green software from requirements, design, and coding to maintenance, from the software development process to the end product, and from making software green to using software to make the world greener.

In "Safety, Security, Now Sustainability: The Nonfunctional Requirement for the 21st Century," authors Birgit Penzenstadler, Ankita Raturi, Debra Richardson, and Bill Tomlinson argue that sustainability (that is, green software) is now considered to be a nonfunctional



requirement similar to safety and security. Their main research question, "Will sustainability requirements become as important as safety and security requirements?" lets the authors explore parallels among these three types of nonfunctional requirements, arguing that some of the existing tools and techniques, applicable in the safety and security areas, can be reused in the sustainability domain. In addition, the authors highlight the importance of indirect impact of the sustainability requirements: an analyst should consider not only direct effects of the requirements (such as energy consumption) but also indirect effects (such as change in consumer behavior).

> Could greened software be less performant, maintainable, and secure, thereby requiring additional trade-offs?

In "Analyzing the Harmful Effect of God Class Refactoring on Power Consumption," Ricardo Pérez-Castillo and Mario Piattini also argue that greenness is in fact a nonfunctional requirement, but they question whether it conflicts with other such requirements. Could greened software be less performant, maintainable, and secure, thereby requiring additional trade-offs? The authors analyze how power consumption changes as an effect of refactoring for maintainability. In this particular case, they found that reducing god classes has a negative effect on consumption.

Green specifications provide a way to indicate a service's carbon footprint and eventually specify operational constraints to allow more flexibility during service provisioning. In "Facilitating Greener IT through Green Specifications," Colin Atkinson, Thomas Schulze, and Sonja Klingert propose a solution that includes environmental aspects as part of a service-level agreement. To test it, they used their approach in an industrial setting in collaboration with a local software-as-a-service provider.

A widely accepted concept about green software is the distinction between greening IT and greening systems through IT. How big is the leverage of one or the other? Precise data about this point is hard to obtain. In "Green Software: Greening What and How Much?," Krzysztof Sierszecki, Tommi Mikkonen, Michaela Steffens, Thomas Fogdal, and Juha Savolainen report the experience and data of a company that develops factory automation systems that can be roughly divided into three parts: computer-controlled drivers, electric motors, and full-fledged industrial applications. The authors analyze the split of consumption (and possible consumption savings) for each part. As expected, greening the IT part brings in a minimal contribution to greening the whole application, so the authors underline the importance of a system approach to addressing the problem.

In the end, most technical systems include humans in the loop, therefore the human factor must be considered for an accurate analysis of technical systems. We all know that cars of the same category can have fairly different levels of fuel efficiency, but that's also a function of the driver's driving style. Does this happen with software systems, too? In "The Impact of User Choice on Energy Consumption," Chenlei Zhang, Abram Hindle, and Daniel M. German take this viewpoint and analyze the energy consumption of software applications in the same category across several usage scenarios. Perhaps not surprisingly, they find that applications do have different efficiencies, so the authors suggest introducing consumption ratings both to make public the efficiency of software applications and to modify user behavior.

Most IT companies have begun to consider green and sustainable strategies to reduce energy cost and contribute to environmental sustainability. However, going forward, they'll need to deal with all the infrastructure requirements and the environmental impact of IT itself (hardware and software) and its use. The challenges of system modernization are immense. Recent developments have indicated that besides focusing on hardware, the IT industry should also focus on software in terms of sustainability. Software development requires making trade-offs between customer demands and the requirements for corporate social responsibility initiatives. In an interview with Steve Raspudic at IBM Toronto's Software Labs, we discussed wide adoption of green software requirements, metrics, and measurement in the industry.

Change is inevitable. Companies need to implement energy-efficient technology services around the globe. This, along with regulations and standards for measuring energy efficiency, will continue to drive the development of energy-efficient pathways. In this context, green software is an ideal way for companies to achieve environmental sustainability and reduce the cost of system and product maintenance.

## Reference


1. G.P. Peters et al., "Rapid Growth in CO2 Emissions after the 2008-2009 Global Financial Crisis," *Nature Climate Change*, vol. 2, 2012, pp. 2–4.


See www.computer.org/software-multimedia for multimedia content related to this article.

## ABOUT THE AUTHORS


**AYSE BASAR BENER** is a professor in and the director of the Data Science Lab at Ryerson University. Her research interests include big data applications to tackle the problem of decision-making under uncertainty and analyzing complex structures in big data to build recommender systems and predictive models in healthcare, software engineering, smart energy grids, and green software. Bener received a PhD in information systems from the London School of Economics. Contact her at ayse.bener@ryerson.ca.



**MAURIZIO MORISIO** is a professor of computer science at Politecnico di Torino, Italy, where he leads the software engineering research group (http://softeng.polito.it). His research interests include software engineering for safety-critical systems, software and system engineering for sociotechnical systems, and smart cities and green ecosystems. Morisio received a PhD in software engineering from Politecnico di Torino. Contact him at maurizio.morisio@polito.it.



**ANDRIY MIRANSKYY** is an assistant professor in the Department of Computer Science at Ryerson University. His research interests include mitigating risk in software engineering, focusing on software quality assurance, big data, and green IT. Miranskyy received a PhD in applied mathematics from the University of Western Ontario. Contact him at avm@ryerson.ca.